DEC STD 070-4 Video Systems Reference Manual - Terminal Management

DOCUMENT IDENTIFIER: A-DS-EL00070-04-0000 Rev A, 04-Oct-1990

ABSTRACT: This section describes device and service-class independent protocols for performing identification and status functions, and for selecting syntax for interchange compatibility.

APPLICABILITY: Mandatory for engineers designing hardware for terminal products and software engineers designing programs using terminal interfaces. Additional requirements are defined in DEC STD 070-1 Video Systems Reference Manual - Concepts and Conformance Criteria.

STATUS: APPROVED 04-Oct-1990; use VTX SMC for current status.

The material contained within this document is assumed to define mandatory standards unless it is clearly marked as: a. not mandatory; or b. guidelines.

Material that is marked as "not mandatory" is considered to be of potential benefit to the corporation and should be followed unless there are good reasons for non–compliance. "Guidelines" define approaches and techniques that are considered to be good practice, but should not be considered as requirements.

This document is confidential and proprietary, and is the property of Digital Equipment Corporation. It is an unpublished work protected under the Federal copyright laws.

©Digital Equipment Corporation. 1991. All rights reserved.

DEC STD 070-4 Video Systems Reference Manual - Terminal Management

DOCUMENT IDENTIFIER: A-DS-EL00070-04-0000 Rev A, 04-Oct-1990

REVISION HISTORY: Rev A 04-Oct-1990

Document Management Category:	Terminal Interface Architecture (STI)
Responsible Department:	VIPS Terminals Architecture
Responsible Person:	Peter Sichel

APPROVAL: This standard, prepared by the Desktop Systems Group, has been reviewed and recommended for approval by the General Review group for its category.

Peter Sichel - VIPS Terminals Architecture

Eric Williams - Standards Process Manager

Direct requests for further information to:

Peter Sichel
Use $ VTX ELF for the latest location information.

Use VTX SMC to order copies of this document from Standards and Methods Control. Send distribution questions to JOKUR::SMC or call DTN: 234-4423.

CONTENTS

1 INTRODUCTION 1
1.1 SCOPE 1
1.2 RELATIONSHIP TO TERMINAL INTERFACE ARCHITECTURE (TIA) 1

2 TERMINOLOGY 2

3 STATE DESCRIPTIONS 3
3.1 DEVICE IDENTIFICATION 3
3.2 TERMINAL STATE DECLARATIONS 5
3.3 STATUS AND TEST 12
3.3.1 Virtual Terminal Configuration 12
3.3.2 Data Integrity 12

4 DEVICE INITIALIZATION 12

5 CONTROL FUNCTIONS 17

6 CHANGE HISTORY 38
6.1 REVISION 0.0 TO 0.1 38
6.2 REVISION 0.1 TO AX10 38
6.3 Rev AX10 to AX11 39
6.4 Rev AX11 to AX12 39

Appendix A REFERENCED DOCUMENTS 43
A.1 EL-Class Digital Documents 43

Appendix B RELATED DOCUMENTS 45
B.1 EL-Class Digital Documents 45
B.2 Other Documents 45
B.3 Ordering Information 45

INDEX

FIGURES

1 Structuring of the Terminal Interface Architecture 1
2 External and Internal Terminal Interfaces 2

TABLES

1 Registered Extensions to the Character Cell Display Service Class 19

1 INTRODUCTION

This standard defines the interface to perform terminal management functions in all terminal devices. Terminal management functions include device control operations that are independent of any display service class and that are not associated with the presentation of data.

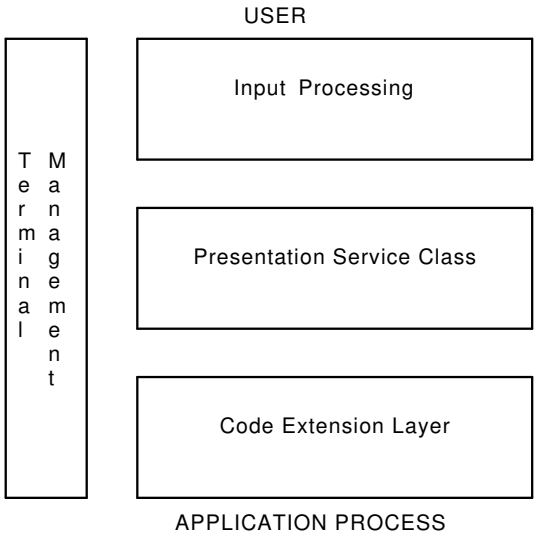
1.1 SCOPE

This standard applies to all Digital products that implement terminal management interfaces and are to be certified as conforming to the Terminal Interface Architecture (TIA). It also applies to software that is intended to use these interfaces in a conforming manner.

1.2 RELATIONSHIP TO TERMINAL INTERFACE ARCHITECTURE (TIA)

The terminal management functions are a unique class of device controls that operate across layers of the TIA.

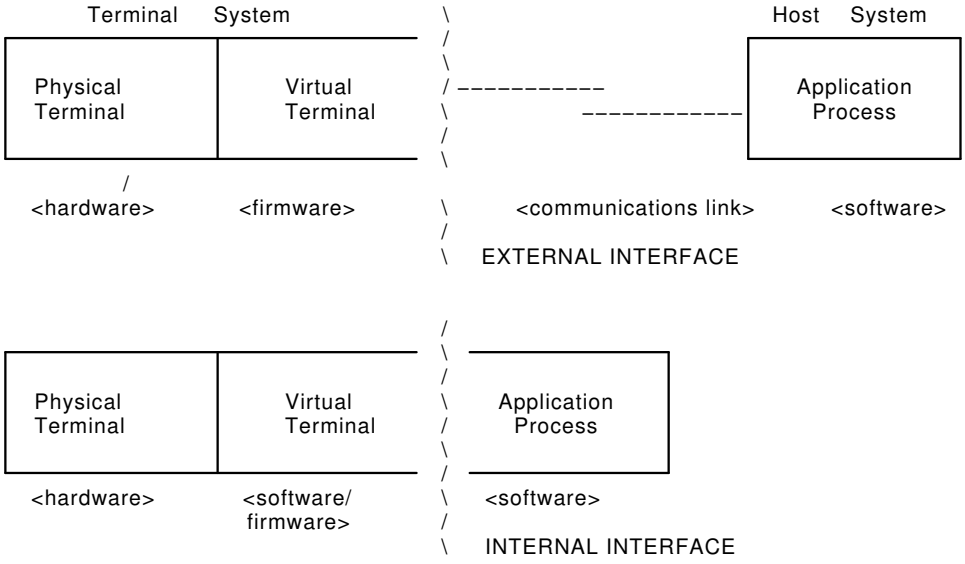
Figure 1: Structuring of the Terminal Interface Architecture



To aid in understanding, functions described in this document are identified as intended for either application, terminal management, or communications use when an external communications interface is present (guideline). These different uses may not always be separated. For example, applications may require their own terminal management. If not otherwise indicated, functions described in this document are intended for terminal management use.

The interfaces defined within this document apply to both internal and external product interfaces. External interfaces are interfaces between a terminal, personal computer, or workstation and a remote system. Internal interfaces are interfaces between a terminal sub-system and software processes running within a terminal, personal computer, or workstation.

Figure 2: External and Internal Terminal Interfaces



2 TERMINOLOGY

Conformance Level - An architectural agreement that defines a number of functions within a class of operations that must be adhered to in all products claiming to implement that level. Conformance levels are defined to provide interface compatibility with other devices at the same level. Selecting a particular conformance level automatically places the terminal into a known state (the Soft Terminal Reset state on video terminals), without affecting the data currently contained in the display.

Device Self-Test - A control provided as a means of invoking device specific diagnostic tests, and subsequently determining the device status by means of the Device Status Report (DSR) control.

Device Status Report - A control used to inquire as to the current state of the terminal device. Note that some Device Status Reports (DSRs) are properly part of terminal management, while others pertain to presentation state or input processing.

Factory Default State - Digital specified default values for all state information. Factory Defaults are also the initial configuration of all setable state when the unit is shipped from the factory (manufacturing site).

Power-Up State - Factory defaults plus any non-volatile settings, such as Non-Volatile Memory (NVM), customization switches. The initial configuration of all setable state immediately following power-up.

Primary Device Attributes Control - The control used to request or report the implemented architectural conformance level and extensions.

Secondary Device Attributes Control - The control used to request or report the identification code of the specific product in which the interface is implemented.

Tertiary Device Attributes Control - The control used to request the identification code of the specific unit in which the interface is implemented.

Terminal Management Functions - A special class of device controls that operate across all layers of the Terminal Interface Architecture (TIA) and are not related to the presentation of data.

3 STATE DESCRIPTIONS

The algorithms in this standard provide a description of terminal state and how it is affected by various functions. They have been compiled for accuracy, but have not been formally certified. They should be viewed the same as other text in the standard.

3.1 DEVICE IDENTIFICATION

A conforming device shall provide an identification mechanism that returns the following information upon request.

a. The general characteristics (display service class) of the device, indicating the architectural level to which the device conforms

b. The architectural extensions that the device implements

c. A product specific code registered by the architecture

d. A revision level code, indicating the release level of the communicating firmware or software implementing the terminal management functions

e. A list, defined by each product implementation, of product specific options installed in the device at the time the identification request is received

f. For Level 4 and higher devices, a unique serial number to identify the specific unit

The following example illustrates the constants used for VT420 terminal identification.

```
/************** Module terminal_identification.h *************/
/*
 * Constants used for terminal identification
 * VT420 example
 */

/*
 * Extensions to the terminal interface architecture
 * Level 1 through Level 4              1-Feb-1990
 */
#define ONE_THIRTY_TWO_COLUMN     1
#define PRINTER_PORT              2
#define REGIS_GRAPHICS            3
#define SIXEL_GRAPHICS            4
#define KATAKANA_EXT              5
#define SELECTIVE_ERASE           6
#define DRCS_EXT                  7
#define UDK_EXT                   8
#define NRCS_EXT                  9
#define KANJI_EXT                10
#define STATUS_DISPLAY_EXT       11
#define SERBO_CROATION           12
#define BLOCK_MODE               13
#define EIGHT_BIT_IA             14
#define TCS                      15    /* Technical Char Set */
#define LOCATOR_PORT             16
#define TSI                      17    /* State Interrogation */
#define WINDOWING                18
#define MULTIPLE_SESSIONS        19    /* via TD/SMP */
#define APL                      20
#define HORIZONTAL_SCROLL        21
#define COLOR_TEXT               22
#define GREEK                    23
#define TURKISH                  24
#define ARABIC_BILINGUAL_M1      25
#define ARABIC_BILINGUAL_M2      26
#define ARABIC_BILINGUAL_M3      27
#define RECTANGULAR_EDITING      28
#define TEXT_LOCATOR             29
#define HANZI                    30
                               /* 31 not assigned */
#define TEXT_MACROS              32
#define HANGUL_HANJA             33
#define ICELANDIC                34
#define ARABIC_BILINGUAL_TC      35
#define ARABIC_BILINGUAL_NOTC    36
#define THAI                     37
#define CHARACTER_OUTLINING      38

#define NUMBER_OF_EXTENSIONS     39

/* device identification */
#define PRODUCT_ID     41     /* VT420 */
#define REVISION_LEVEL 10     /* Version 1.0 times 10 */

/* hardware options */
#define NUMBER_OF_OPTIONS 0   /* VT420 has no options */
```

3.2 TERMINAL STATE DECLARATIONS

The code example below illustrates type declarations for the terminal state. Type declarations in C do not allocate any storage. All these types can be defined regardless of the conformance level or extensions actually implemented.

```
/**************  Module terminal_state_types.h  *************/
/*
 * type declarations for terminal state       PAS 1-Feb-1990
 * VT420 example
 */

/*
 * terminal management
 */
#include "terminal_identification.h"    /* DA definitions */
typedef short int conformance_level_t;
#define LEVEL_1 1
#define LEVEL_2 2
#define LEVEL_3 3
#define LEVEL_4 4

typedef enum
    {           /* device_status_t */
    READY,
    MALFUNCTION
    } device_status_t;

typedef enum { LIMITED, UNLIMITED } transmit_rate_limit_mode_t;
typedef enum
    {           /* vt_configuration_t */
    SINGLE_SESSION,
    SESSIONS_ON_COMM,
    S1_HOST_S2_PRINTER,
    S2_HOST_S1_PRINTER
    /* VT330 with 2nd Host Port
    SESSIONS_ON_COMM2,
    S1_COMM1_S2_COMM2,
    S2_COMM1_S1_COMM2  */
    } vt_configuration_t;

typedef enum
    {          /* control_representation_mode_t */
    INTERPRET_CONTROLS,
    DISPLAY_CONTROLS
    } control_representation_mode_t;

/*
 * Code Extension Layer
 */
/* type declarations */
#include "gparse.h"    /* ANSI parser definitions */
typedef enum { ASCII_C }         c0_control_set_t;
typedef enum { SUPPLEMENTAL_C }  c1_control_set_t;
typedef enum
    {           /* graphic_character_set_t */
    ASCII_G,
    LINE_DRAWING_G,
    DEC_SUPPLEMENTAL_G,
    ISO_LATIN1_SUPPLEMENTAL_G,
    DRCS_G,
    NRCS_G,
    UPSS_G,
    TCS_G
```

```
    } graphic_character_set_t;
typedef enum { G0, G1, G2, G3 } gset_name_t;
typedef struct
    {           /* in_use_table_t */
    c0_control_set_t           c0;
    graphic_character_set_t    gl;
    gset_name_t                invoked_gl;

    c0_control_set_t           c1;
    graphic_character_set_t    gr;
    gset_name_t                invoked_gr;
    } in_use_table_t;
typedef enum { NO_SINGLE_SHIFT, SS2, SS3 } single_shift_t;
typedef enum
    {        /* in_control_string_t */
    NOT_IN_CONTROL_STRING,
    REGIS,
    SIXEL,
    DRCS_LOAD,
    UDK_LOAD,
    MACRO_LOAD,
    TERMINAL_STATE_LOAD,
    PRESENTATION_STATE_LOAD,
    SET_TERMINAL_UNIT_ID
    } in_control_string_t;
typedef enum { EIGHT_BIT, SEVEN_BIT } environment_t;
typedef enum { EIGHT_BIT_MULTINATIONAL, SEVEN_BIT_NATIONAL }
    character_set_mode_t;

#include "char_cell_types.h"    /* character cell display */

/*
 * Status Display Extension
 */
typedef enum { MAIN_DISPLAY, STATUS_DISPLAY } active_display_t;
typedef enum { NO_STATUS_DISPLAY, INDICATOR, HOST } status_display_t;

/*
 * keyboard processing
 */
typedef enum { UNLOCKED, LOCKED } keyboard_action_mode_t;
typedef enum { REPEAT_ON, REPEAT_OFF } auto_repeat_mode_t;
typedef enum { CURSOR, CK_APPLICATION } cursor_key_mode_t;
typedef enum { NUMERIC, KP_APPLICATION } keypad_mode_t;
typedef enum { TYPEWRITER, DATA_PROCESSING }
    keyboard_usage_mode_t;
typedef enum { CAPS_LOCK, SHIFT_LOCK } caps_shift_lock_mode_t;
typedef enum { HOLD_OFF, HOLD_ON } hold_screen_mode_t;
typedef enum { LOCK_OFF, LOCK_ON } lock_mode_t;
typedef enum { KEYCLICK_OFF, KEYCLICK_ON } keyclick_mode_t;
typedef enum { ECHO_ON, ECHO_OFF } send_receive_mode_t;
typedef enum { DELETE, BACKSPACE } backarrow_key_mode_t;
typedef enum { LOCAL, REPORT, IGNORE } local_key_mode_t;
typedef enum { LOCAL_F5, SHIFTED_ONLY, REPORT_F5, IGNORE_F5 }
    F5_key_mode_t;
typedef enum { CHARACTER, POSITION } keyboard_encoding_mode_t;
typedef short int keyboard_dialect_t;
#define KEYBOARD_UNDEFINED 0
#define NORTH_AMERICAN     1
#define BRITISH            2
#define FLEMISH            3
#define CANADIAN_FRENCH    4
#define DANISH             5
#define FINNISH            6
#define GERMAN             7
#define DUTCH              8
```

```
#define ITALIAN            9
#define SWISS_FRENCH      10
#define SWISS_GERMAN      11
#define SWEDISH           12
#define NORWEGIAN         13
#define BELGIAN_FRENCH    14
#define SPANISH           15
#define PORTUGUESE        16
#define CANADIAN_ENGLISH  17
#define MAX_NUM_DIALECTS  18

/*
 * printer port
 */
typedef enum { PRINTER_CONTROLLER_OFF, PRINTER_CONTROLLER_ON }
    printer_controller_mode_t;
typedef enum { AUTO_PRINT_OFF, AUTO_PRINT_ON } auto_print_mode_t;
typedef enum { PRINT_FF_OFF, PRINT_FF_ON } print_form_feed_mode_t;
typedef enum { PRINT_SCROLLING_REGION, PRINT_DISPLAY }
    print_extent_mode_t;
typedef enum { NO_PRINTER_TO_HOST, PRINTER_TO_HOST }
    printer_to_host_mode_t;
typedef enum { SHARED, SESSION1, SESSION2 }
    printer_assignment_t;
typedef enum { NATIONAL_ONLY, NATIONAL_PLUS_LINE_DRAWING,
               MULTINATIONAL_ONLY, ALL_CHARACTERS } printer_style_t;

/*
 * Graphics
 */
typedef enum { GRAPHICS_CURSOR_ON, GRAPHICS_CURSOR_OFF }
    graphics_cursor_enable_mode_t;    /* ReGIS */



/**************  module char_cell_types.h  *************/
/*
 * type declarations for character cell display state
 * PAS 28-Jul-1989
 */

typedef enum { JUMP, SLOW } scrolling_mode_t;
typedef enum { NORMAL_SCREEN, REVERSE_SCREEN } screen_mode_t;
typedef enum { WRAP_OFF, WRAP_ON } auto_wrap_mode_t;
typedef enum { TEXT_CURSOR_OFF, TEXT_CURSOR_ON }
    text_cursor_enable_mode_t;
typedef enum { INSERT, REPLACE } insert_replace_mode_t;
typedef enum { NEW_LINE_OFF, NEW_LINE_ON } new_line_mode_t;
typedef short int line_t;
typedef short int column_t;
typedef short int page_t;
typedef struct
    {         /* character_position_t */
    line_t line;
    column_t column;
    page_t page; /* L4 or Windowing */
    } character_position_t;
typedef enum { TRUE, FALSE } boolean_t;
typedef struct
    {         /* character_rendition_t */
    boolean_t bold;
    boolean_t blink;
    boolean_t underscore;
    boolean_t reverse;
    } character_rendition_t;
typedef struct
```

```
    {        /* character_attribute_t */
    boolean_t selective_erase;
    } character_attribute_t;
typedef enum
    {        /* line_rendition_t */
    DOUBLE_HEIGHT_TOP,
    DOUBLE_HEIGHT_BOTTOM,
    SINGLE_WIDTH,
    DOUBLE_WIDTH
    } line_rendition_t;
typedef struct
    {       /* character_t */
    short int code;
    character_rendition_t rendition;
    character_attribute_t attribute;
    graphic_character_set_t character_set;
    } character_t;
typedef struct
    {       /* char_set_designator_t */
    graphic_character_set_t name;
    short int first_intermediate;    /* 0-47, 0=no intermediate */
    short int second_intermediate;   /* 0-47 */
    short int final;                 /* 0 indicates null entry */
    } char_set_designator_t;
typedef enum { ABSOLUTE, DISPLACED } origin_mode_t;
typedef struct
    {      /* save_buffer_t */
    character_position_t position;
    character_rendition_t rendition;
    origin_mode_t origin_mode;
    graphic_character_set_t left;
    graphic_character_set_t right;
    graphic_character_set_t g0;
    graphic_character_set_t g1;
    graphic_character_set_t g2;
    graphic_character_set_t g3;
    character_attribute_t attribute;
    } save_buffer_t;
typedef enum { EIGHTY, ONE_THIRTY_TWO } column_mode_t;
typedef enum { UNCOUPLED, COUPLED } cursor_coupling_mode_t;
typedef enum { FIXED, SETABLE } left_right_margins_mode_t;
typedef enum { STREAM, RECTANGULAR } area_extent_t;
typedef short int lines_per_page_t;    /* 24, 25, 36, 72, 144 */
typedef short int columns_per_page_t;  /* 80, 132 */
typedef short int number_of_pages_t;   /* 1, 2, 3, 4, 5, 6 */
/************ end of char_cell_types.h  ****************/
```

The following example illustrates storage declarations for the terminal state. Only those state variables required for the desired conformance level and extensions need be present.

```
/**************  Module terminal_state_extern.h  *************/
/*
 * external storage declarations for terminal state
 * VT420 example                    PAS 28-Feb-1989
 */

/*
 * This code can also used to allocate storage
 * by redefining "ext" to be null.
 */
#define ext extern

#include "terminal_state_types.h"

/*
 * terminal management
 */
ext conformance_level_t conformance_level;
ext boolean_t level_1_extensions[NUMBER_OF_EXTENSIONS];
ext boolean_t level_2_extensions[NUMBER_OF_EXTENSIONS];
ext boolean_t level_3_extensions[NUMBER_OF_EXTENSIONS];
ext boolean_t level_4_extensions[NUMBER_OF_EXTENSIONS];
/* VT420 has no options --
ext boolean_t product_options[NUMBER_OF_OPTIONS];
--*/
ext long int terminal_unit_id;    /* Level 4, VT420 example */
ext device_status_t device_status;
ext transmit_rate_limit_mode_t transmit_rate_limit_mode;
ext boolean_t power_up_detected_flag;    /* Level 4 */
ext boolean_t error_detected_flag;       /* Level 4 */
ext vt_configuration_t vt_configuration; /* session mgmt ext */
ext control_representation_mode_t
        control_representation_mode;     /* Set-Up */
ext boolean_t line_local_mode;           /* Set-Up */

/*
 * Code Extension Layer
 */
ext struct parse_stack comm_state;      /* ANSI parser state */
ext graphic_character_set_t  designated_graphic_sets[4];
ext in_use_table_t           in_use_table;
ext single_shift_t           single_shift;
ext in_control_string_t      in_control_string;
ext graphic_character_set_t  upss;    /* Level 3 or 8-bit IA */
ext character_set_mode_t     character_set_mode;
ext character_set_mode_t     r_character_set_mode;
ext environment_t
    host_port_environment,
    c1_transmission_mode;

#include "char_cell_extern.h"    /* character cell display */
```

```
/*
 * keyboard processing
 */
ext keyboard_action_mode_t  keyboard_action_mode;
ext auto_repeat_mode_t      auto_repeat_mode;
ext cursor_key_mode_t       cursor_key_mode;
ext keypad_mode_t           keypad_mode;
ext keyboard_usage_mode_t   keyboard_usage_mode;     /* Level 3 */
ext caps_shift_lock_mode_t  caps_shift_lock_mode;
ext hold_screen_mode_t      hold_screen_mode;
ext lock_mode_t             lock_mode;
ext keyclick_mode_t         keyclick_mode;
ext send_receive_mode_t     send_receive_mode;
ext send_receive_mode_t     r_send_receive_mode;
ext backarrow_key_mode_t    backarrow_key_mode;      /* Level 3 */
ext local_key_mode_t                                 /* Level 4 */
        compose_key_mode,
        alt_key_mode,
        F1_key,
        F2_key,
        F3_key,
        F4_key;
ext F5_key_mode_t           F5_key_mode;             /* Level 4 */
ext keyboard_encoding_mode_t keyboard_encoding_mode; /* Level 4 */
ext keyboard_dialect_t      keyboard_dialect;           /* NRCS */
ext graphic_character_set_t nrcs_list[MAX_NUM_DIALECTS];/* NRCS */

/*
 * printer port extension
 */
ext printer_controller_mode_t  printer_controller_mode;
ext auto_print_mode_t          auto_print_mode;
ext print_form_feed_mode_t     print_form_feed_mode;
ext print_extent_mode_t        print_extent_mode;
ext printer_to_host_mode_t     printer_to_host_mode; /* Level 3 */
ext printer_assignment_t       printer_assignment;  /* sessions */
ext environment_t              printer_port_environment;
ext printer_style_t            printer_style;

/*
 * Graphics
 */
ext graphics_cursor_enable_mode_t
        graphics_cursor_enable_mode;   /* ReGIS */

/*
 * communications layer
 */
ext int host_port;     /* logical unit number for comm port */
ext int printer_port;

/**************  module char_cell_extern.h  *************/
/*
 * external storage declarations for character cell display state
 * PAS 28-Jul-1989
 */
```

```
#define MAX_NUM_LINES 73
#define MAX_NUM_COLUMNS 133
#define EMPTY_CHARACTER 0
#define MAX_NUM_CHAR_SETS 8
/* character set table entries */
#define TABLE_ASCII 0
#define TABLE_LINE_DRAWING 1
#define TABLE_DEC_SUPP 2
#define TABLE_NRCS 3
#define TABLE_LATIN1 4
#define TABLE_UPSS 5
#define TABLE_TCS 6
#define TABLE_DRCS 7
/* Note DRCS must be last in search order
   to replace others if identical */

ext character_t            display[MAX_NUM_LINES][MAX_NUM_COLUMNS];
ext line_rendition_t       line_rendition[MAX_NUM_LINES];
ext character_position_t   active_position;
ext line_t                 top_margin;
ext line_t                 bottom_margin;
ext column_t               left_margin;    /* Horiz scroll ext */
ext column_t               right_margin;   /* Horiz scroll ext */
ext boolean_t              last_column_flag;
ext active_display_t       active_display; /* L3 or Status Dsply*/
ext status_display_t       status_display; /* L3 or Status Dsply*/
ext char_set_designator_t  char_set_table[MAX_NUM_CHAR_SETS];
ext save_buffer_t          cursor_save_buffer;
ext character_rendition_t  current_rendition;
ext character_attribute_t  current_attribute; /* Selective Erase*/
ext boolean_t              horizontal_tab_stops[MAX_NUM_COLUMNS];
ext column_mode_t          column_mode;      /* 132 Column Ext */
ext lines_per_page_t       lines_per_page;   /* Windowing */
ext columns_per_page_t     columns_per_page; /* Windowing */
ext number_of_pages_t      number_of_pages;  /* Windowing */
ext scrolling_mode_t       scrolling_mode;
ext screen_mode_t          screen_mode;
ext origin_mode_t          origin_mode;
ext auto_wrap_mode_t       auto_wrap_mode;
ext text_cursor_enable_mode_t  text_cursor_enable_mode;
ext insert_replace_mode_t  insert_replacement_mode;
ext new_line_mode_t        new_line_mode;
ext cursor_coupling_mode_t
        horizontal_cursor_coupling,      /* Windowing */
        vertical_cursor_coupling,        /* Windowing */
        page_cursor_coupling;            /* Windowing */
ext left_right_margins_mode_t
        left_right_margins_mode; /* Horiz Scroll */
ext area_extent_t
        attribute_change_extent; /* L4 or rect editing */
/********** end of char_cell_extern.h  *************/
```

3.3 STATUS AND TEST

Upon request, a conforming device shall provide at the terminal interface a means of returning device status information. Status information consists of either "Device Ready" or "Device Malfunction". (See subhead 5, DEVICE STATUS REPORT.) The device status shall be set upon power-up initialization or receipt of RESET TO INITIAL STATE (RIS). See *Video Systems Reference Manual - Documented Exceptions*.

The architecture does not define the nature or duration of specific device test sequences. The architecture provides a registry of test invocation control functions (Refer to *DEC STD 138-0 Registry of Control Functions for Character-Imaging Devices*) that may have different effects on different products. However, failure of any applicable test sequences shall set the device status to the "Device Malfunction" state.

3.3.1 Virtual Terminal Configuration

If the Multiple Sessions extension is supported, a conforming device shall provide a means at the terminal interface of returning information on whether the terminal is configured to support virtual terminals. Status information returned shall be in the form of a Device Status Report (DSR) (see subhead 5, Device Status Report) and consist of: "Virtual Terminals Not Enabled", "Virtual Terminals Enabled Over Physical Lines", "Virtual Terminals Available Through TD/SMP".

This feature is intended for terminal management use. Specific information on the state of the Terminal Device/Session Management Protocol (TD/SMP) or other session management protocol should be ascertained through the protocol handler.

3.3.2 Data Integrity

At Conformance Level 4, a conforming device shall provide a means at the terminal interface of returning the status of a data integrity flag. Status information returned shall be in the form of a Device Status Report (DSR) (see subhead 5 Device Status Report) and consist of: "OK, no comm error detected since last report", "Error, a comm error has been detected since last report", or "Power-up, No request since last power-up or reset".

This is an application feature for defensive programming, and not intended for terminal management or communications use.

4 DEVICE INITIALIZATION

On power up, all state information (state) in the terminal shall be set to a known value, or value previously selected by the user and stored through non-volatile means. This state is referred to as the Power-Up State.

Since the architecture allows the Power-Up State of certain features to be set and stored under local control (Set-Up and non-volatile memory for example), software cannot rely on the values of this state after an appropriate initialization sequence is executed or after the device is powered up. The values given in the algorithm below apply only to those implementations in which the state cannot be set and stored locally by the terminal user. The state defined by the architecture to which this indeterminate condition may apply is limited to the following:

```
conformance_level

host_port_environment
vt_configuration
c1_transmission_mode
transmit_rate_limit_mode
terminal_unit_id
character_set_mode
upss
columns_per_page
lines_per_page
number_of_pages
status_display

column_mode
scrolling_mode
screen_mode
auto_wrap_mode
text_cursor_enable_mode
new_line_mode
horizontal_tab_stops

vertical_cursor_coupling
horizontal_cursor_coupling
page_cursor_coupling

auto_repeat_mode
cursor_key_mode
caps_shift_lock_mode

keyboard_usage_mode
backarrow_key_mode
alt_key_mode
compose_key_mode
F1_key_mode
F2_key_mode
F3_key_mode
F4_key_mode

auto_print_mode

printer_controller_mode

print_form_feed_mode

print_extent_mode
printer_to_host_mode
printer_style

graphics_cursor_enable_mode
UDK definitions (see DEC-STD-070-11, UDK Extension)
```

State Affected:

All terminal state. See subhead 3.2, Terminal State Declarations.

The following code example illustrates terminal management routines.

Algorithm:

```
/**************  Module terminal_management.c  *************/
/*
* terminal management routines
* VT420 example
*/

/* get access to global terminal state */
#include "terminal_state_extern.h"
```

```
/* function declarations */
void hardware_init();
void comm_init();
void setup_init();
void host_state_init();
void soft_terminal_reset();
int send_char();
int send_string();
int send_int();
int send_hex();

void power_up_initialization()
{
hardware_init();     /* actual hardware as needed */
comm_init();         /* communications layer */
setup_init();        /* features that are not host setable */
host_state_init();   /* all host setable state */
}   /* end procedure power_up_initialization */

void setup_init()
/* initialize local features that are not host setable */
{
host_port_environment = EIGHT_BIT;     /* NVM */
vt_configuration = SINGLE_SESSION;     /* NVM */
control_representation_mode =
INTERPRET_CONTROLS;
printer_port_environment = EIGHT_BIT;  /* NVM */
printer_style = NATIONAL_ONLY;         /* NVM */
}   /* end procedure setup_init */

void host_state_init()
{
line_t   x;
column_t y;
/*.................*/

/* do initialization */

/* Terminal Management */
conformance_level = LEVEL_4;           /* NVM */
transmit_rate_limit_mode = LIMITED;    /* NVM */
power_up_detected_flag = TRUE;
error_detected_flag = FALSE;
vt_configuration = SINGLE_SESSION;     /* NVM */
/* VT420 terminal unit ID example (different for each unit) */
terminal_unit_id = 0x01000001;   /* HKO unit 1,  NVM */

/* Code Extension */
upss = DEC_SUPPLEMENTAL_G;             /* NVM */

in_use_table.c0 = ASCII_C;
in_use_table.gl = ASCII_G;
in_use_table.invoked_gl = G0;

in_use_table.c1 = SUPPLEMENTAL_C;
in_use_table.gr = upss;
in_use_table.invoked_gr = G2;

designated_graphic_sets[0] = ASCII_G;
designated_graphic_sets[1] = ASCII_G;
```

```
switch (conformance_level)
    {
     case LEVEL_1:
      in_use_table.gr = ASCII_G;
      designated_graphic_sets[2] = ASCII_G;
      designated_graphic_sets[3] = ASCII_G;
      c1_transmission_mode = SEVEN_BIT;
      character_set_mode = SEVEN_BIT_NATIONAL;
      break;
     case LEVEL_2:
     case LEVEL_3:
     case LEVEL_4:
      designated_graphic_sets[2] = upss;
      designated_graphic_sets[3] = upss;
      c1_transmission_mode = EIGHT_BIT;              /* NVM */
      character_set_mode = EIGHT_BIT_MULTINATIONAL;  /* NVM */
    };

parse_init(comm_state);  /* initialize ANSI parser */
single_shift = NO_SINGLE_SHIFT;
in_control_string = NOT_IN_CONTROL_STRING;

host_port_environment = EIGHT_BIT;              /* NVM */

/* character cell display */
for (y=1; y<MAX_NUM_LINES; y++)
    {
    for (x=1; x<MAX_NUM_COLUMNS; x++)
     {
     display[y][x].code = EMPTY_CHARACTER;
     }
    line_rendition[y] = SINGLE_WIDTH;
    }
column_mode = EIGHTY;                           /* NVM */
scrolling_mode = SLOW;                          /* NVM */
screen_mode = NORMAL_SCREEN;                    /* NVM */
origin_mode = ABSOLUTE;
auto_wrap_mode = WRAP_OFF;                      /* NVM */
text_cursor_enable_mode = TEXT_CURSOR_ON;       /* NVM */
insert_replacement_mode = REPLACE;              /* NVM */
new_line_mode = NEW_LINE_OFF;                   /* NVM */
active_display = MAIN_DISPLAY;
status_display = NO_STATUS_DISPLAY;             /* NVM */
horizontal_cursor_coupling = UNCOUPLED;         /* NVM */
vertical_cursor_coupling = COUPLED;             /* NVM */
page_cursor_coupling = COUPLED;                 /* NVM */
left_right_margins_mode = FIXED;
attribute_change_extent = STREAM;
lines_per_page = 24;                            /* NVM */
number_of_pages = MAX_NUM_LINES/lines_per_page;

active_position.line = 1;
active_position.column = 1;
active_page = 1;
top_margin = 1;
bottom_margin = lines_per_page;
left_margin = 1;
/* if Windowing Extension,
    columns_per_page in NVM will override column_mode */
if (column_mode == EIGHTY) columns_per_page = 80;  /* NVM */
    else columns_per_page = 132;
right_margin = columns_per_page;
```

```
current_rendition.bold       = FALSE;
current_rendition.blink      = FALSE;
current_rendition.underscore = FALSE;
current_rendition.reverse    = FALSE;
current_attribute.selective_erase = FALSE;

horizontal_tab_stops[1] = FALSE;                /* NVM */
for (x=2; x<MAX_NUM_COLUMNS; x++)
    {
    if (x%8 == 1) horizontal_tab_stops[x] = TRUE;
    else horizontal_tab_stops[x] = FALSE;
    }

save_cursor();  /* initialize cursor save buffer */
macro_init();   /* initialize text macros (Level 4) */

/* keyboard processing */
keyboard_action_mode = UNLOCKED;
auto_repeat_mode = REPEAT_ON;               /* NVM */
cursor_key_mode = CURSOR;                   /* NVM */
keypad_mode = NUMERIC;
keyboard_usage_mode = TYPEWRITER;           /* NVM */
caps_shift_lock_mode = CAPS_LOCK;           /* NVM */
hold_screen_mode = HOLD_OFF;
lock_mode = LOCK_OFF;
keyclick_mode = KEYCLICK_ON;                /* NVM */
send_receive_mode = ECHO_OFF;               /* NVM */
backarrow_key_mode = DELETE;                /* NVM */
compose_key_mode = LOCAL;                   /* NVM */
alt_key_mode = REPORT;                      /* NVM */
F1_key = LOCAL;                             /* NVM */
F2_key = LOCAL;                             /* NVM */
F3_key = LOCAL;                             /* NVM */
F4_key = LOCAL;                             /* NVM */
F5_key_mode = LOCAL_F5;                     /* NVM */
keyboard_encoding_mode = CHARACTER;

/* printer port */
printer_port_environment = EIGHT_BIT;       /* NVM */
printer_controller_mode =
      PRINTER_CONTROLLER_OFF;                /* NVM */
auto_print_mode = AUTO_PRINT_OFF;           /* NVM */
print_form_feed_mode = PRINT_FF_OFF;        /* NVM */
printer_to_host_mode = NO_PRINTER_TO_HOST; /* NVM */
print_extent_mode = PRINT_DISPLAY;          /* NVM */
printer_assignment = SHARED;

/* initialize other extensions */
graphics_cursor_enable_mode = GRAPHICS_CURSOR_ON; /* NVM */
udk_init();                                 /* NVM */
drcs_init();

/* terminal management */
device_status = READY;

};    /* end procedure host_state_init */
```

5 CONTROL FUNCTIONS

DEVICE ATTRIBUTES (Primary) **DA**

Levels: 1x, 2, 3, 4

Purpose: Request or report the device attributes.

Request Format:

```
CSI          Ps          c                              default Ps: 0
9/11         Ps          6/3
```

Response Format:

```
CSI          ?           Ps          c
9/11         3/15        Ps          6/3
```

Description: Use the primary Device Attributes (DA) control to request or report the implemented architectural conformance level and extensions. An omitted parameter or a parameter value of zero (0) indicates a request for identification. The response to the primary DA request consists of request for a DA control sequence using a private parameter string with the introducer 3/15 (?) followed by a variable number of parameters.

The first parameter in the string indicates the basic service class of the device and the architectural level to which the device conforms. The rest of the parameters indicate the architectural extensions supported at the current operating level.

NOTE

Exceptions to the architecture that are documented as appendices to these standards are not reported in the primary DA response.

The value of the first parameter is encoded so that a simple range check can be performed to determine the basic service class and level of the device. Each service class is designated a range of decimal values from 1 to 9. For example, Character Cell Display devices are in the range 61 to 69. The architectural level to which the device conforms is indicated by the low order digit. Thus, a response with a first parameter of 61 indicates that the device conforms to Level 1 of the Character Cell Display architecture. Higher levels, by definition, support the functionality of lower levels of the same service class.

Therefore, if a device code of 63 is returned, the device may be supported as a Level 1, Level 2, or Level 3 Character Cell Display device. This does not mean higher conformance levels are strict supersets of lower conformance levels, but that a device that supports a higher conformance level must support the lower levels as well (using DECSCL).

In general, higher conformance levels are supersets of lower levels, but specific control functions can be interpreted differently to accommodate evolving requirements. Conforming software shall select a known conformance level to guarantee compatibility. See the description of Select Conformance Level (DECSCL) in this document for further information.

Note that the device code in the response represents the highest level that the device can support, and does not indicate the level at which the device is currently operating. For example, a Level 3 conforming device operating at Level 2, as selected by the Select Conformance Level control function, would still respond with a device code of 63.

Table 1 lists extensions to the character cell display service class registered within Digital that may be transmitted by a device in response to an appropriate request. Levels 1 - 4 (DA parameters 61 - 64) have been defined for the character cell display service class.

Table 1: Registered Extensions to the Character Cell Display Service Class

Extension Parameter	Levels	Extension Name
1	1 - 4	132 Column Display
2	1 - 4	Printer Port
3	1 - 4	ReGIS Display
4	1 - 4	Sixels Display
5	1 - 3	Katakana
6	2 - 4	Selectively Erasable Characters
7	2 - 4	Dynamically Redefinable Character Sets
8	2 - 4	User Defined Keys
9	1 - 4	National Replacement Character Sets
10	1 - 3	Kanji
11	2	Status Display
12	1 - 3	Serbo-Croation
13	1 - 3	Block Mode
14	2	8-bit Interface Architecture
15	2 - 4	Technical Character Set
16	1 - 4	Locator Port
17	2	Terminal State Interrogation
18	2 - 4	Windowing
19	2 - 4	Multiple Sessions (TD/SMP)
20	1 - 4	APL
21	2 - 4	Horizontal Scrolling
22	2 - 4	Color Text
23	1 - 3	Greek
24	1 - 3	Turkish
25	2, 3	Arabic Bilingual Mode 1
26	2, 3	Arabic Bilingual Mode 2
27	2, 3	Arabic Bilingual Mode 3
28	2, 3	Rectangular Editing
29	2 - 4	Text Locator
30	2, 3	Hanzi
31		
32	2 - 4	Text Macros
33	2, 3	Hangul and Hanja
34	2, 3	Icelandic
35	2, 3	Arabic Bilingual with Text Controls
36	2, 3	Arabic Bilingual with no Text Controls
37	2, 3	Thai
38	2, 3	Character Outlining

Implicit Extensions

Implicit Extensions are features originally defined at a higher conformance level that still operate when a lower conformance level is selected and are not required to be explicitly reported. The "8-bit Interface Architecture" extension, for example, was originally defined in Level 3 and as an extension to Level 2. A Level 3 device operating at Level 2 need not disable the 8-bit Interface Architecture Extension, nor report it explicitly as a parameter in the DA response. Level 3 or higher defines 8-bit Interface Architecture to be an implicit extension to Level 2.

Each conformance level may define a set of implicit extensions at lower levels. Higher conformance levels must support all the implicit extensions defined by lower levels. Implicit extensions must only be allowed if they are fully backward compatible with the lower conformance level.

Software can determine whether an implicit extension is present by seeing if the reported conformance level is greater than or equal to the conformance level at which the feature was originally defined.

If a higher conformance level subsumes extensions previously reported at a lower level, these extensions shall be reported at the lower level so that existing software can determine if the extension is present.

Defining implicit extensions allows features defined by a higher conformance level to operate at a lower conformance level if they are consistent with that level.

Level Reported	Operating Level	Implicit Extension
2 or higher	1	DECSCL
3 or higher	2	11 Status Display
	2	14 8-bit Interface Architecture
	2	17 Terminal State Interrogation
4 or higher	1-3	DECSR

State Affected: None

Algorithm:

```
void device_attributes_1()
{
int n;
send_char(0x9B, host_port);
send_string("?64", host_port);
if (conformance_level == LEVEL_1)
    for (n=1; n<NUMBER_OF_EXTENSIONS; n++)
     if (level_1_extensions[n] == TRUE)
      {
      send_char(';', host_port);
      send_int(n, host_port);
      };
if (conformance_level == LEVEL_2)
    for (n=1; n<NUMBER_OF_EXTENSIONS; n++)
     if (level_2_extensions[n] == TRUE)
      {
      send_char(';', host_port);
      send_int(n, host_port);
      };
if (conformance_level == LEVEL_3)
    for (n=1; n<NUMBER_OF_EXTENSIONS; n++)
     if (level_3_extensions[n] == TRUE)
      {
      send_char(';', host_port);
      send_int(n, host_port);
      };
if (conformance_level == LEVEL_4)
    for (n=1; n<NUMBER_OF_EXTENSIONS; n++)
     if (level_4_extensions[n] == TRUE)
      {
      send_char(';', host_port);
      send_int(n, host_port);
      };
send_char('c', host_port);
};
```

Known Deviations:

Terminals designed before this architecture was developed do not conform to the above encoding scheme. They will, however, respond to the Device Attributes request by returning a DA control with a variable length parameter string. The first parameter of this parameter string is a device-specific code in the range 1 to 59. Thus, these devices are easily distinguished from those that implement the architectural response, whose first parameter is always 60 or greater.

Some terminals provide an alias ID (identification) capability, selectable under local control, to emulate the DA response of older terminals. The factory default will always be to respond with the correct identification code for the family of devices in which the interface is implemented.

The Greek and Turkish special versions of the VT200 and VT300 series use extension parameters 10 and 11 instead of the registered parameter values 23 and 24, respectively.

SELECT CONFORMANCE LEVEL DECSCL

Purpose: Select a particular level of operation for interface compatibility.

Levels: 1x, 2, 3, 4

Format:

```
CSI         Ps          ;          Ps          "          p
9/11        Ps          ;          Ps          2/2        7/0
```

Description: The Select Conformance Level (DECSCL) control selects a terminal operating mode that provides interface compatibility defined by a conformance level of the architecture. The operations and state information available in each level are fully defined in the architectural specifications. The first parameter selects the conformance level of operation. The optional second parameter applies only to Level 2 (it will be ignored in Level 1 operation). The second parameter may be used to select the C1 control transmission for 7-bit or 8-bit controls, in accordance with the following table:

```
0 - default (8-bit C1 controls)
1 - 7-bit C1 controls
2 - 8-bit C1 controls
```

NOTES

1. **The DECSCL control function is required only in Level 2. However, in Level 2 (and higher) conforming devices, the DECSCL control must also be executed when the device is in Level 1 operation. This is an exception to the normal rule that Level 2 functions only work in Level 2 operation, and is necessary to enable switching between levels. DECSCL is an Implicit Extension to Level 1.**
2. **At the present time, four conformance levels are defined by the Terminal Interface Architecture (TIA):**
   61 - Level 1 Character Cell Display Terminals
   62 - Level 2 Character Cell Display Terminals
   63 - Level 3 Character Cell Display Terminals
   64 - Level 4 Character Cell Display Terminals

   Software should be designed to anticipate future higher-numbered levels that will be supersets of Level 1-4.
3. **Selecting a particular conformance level also causes a "soft reset" of the terminal to be performed. This automatically places the terminal into a known state, without affecting the data contained in the display. See Soft Terminal Reset, DECSTR, for more information.**
4. **Attempts to select conformance levels that are unsupported by the device will be ignored.**

State Affected:
conformance_level
c1_transmission_mode

Algorithm:

```
void select_conformance_level(service_class, c1_mode)
    int service_class;
    int c1_mode;
{
switch (service_class)
    {
    case 61:
     conformance_level = LEVEL_1;
     break;
    case 62:
     conformance_level = LEVEL_2;
     break;
    case 63:
     conformance_level = LEVEL_3;
     break;
    case 64:
     conformance_level = LEVEL_4;
     break;
    default: return;
    };
soft_terminal_reset();
switch (c1_mode)
    {
    case 0:
    case 2:
     c1_transmission_mode = EIGHT_BIT;
     break;
    case 1:
     c1_transmission_mode = SEVEN_BIT;
    };
   };    /* end procedure select_conformance_level() */
```

Known Deviations: None

DEVICE ATTRIBUTES (Secondary) DA2

Levels: 1X, 2, 3, 4

Purpose: Request or report the device identification.

Request Format:

CSI	>	Ps	c	Default Ps: 0
9/11	3/14	Ps	6/3	

Response Format:

CSI	>	Ps	c
9/11	3/14	Ps	6/3

Description: The DA2 control requests or reports the identification code of the specific product in which the interface is implemented. The request consists of a DA control sequence with the private parameter code 3/14 (>) and an omitted parameter or a parameter value of zero (0).

The response to the DA2 request consists of a DA control sequence with the private parameter code 3/14 (>) followed by a variable number of parameters. The first parameter indicates the product specific identification of the device. This parameter value shall be registered with the architecture.

The second parameter indicates the revision level of the firmware or software implementing the terminal management functions. Revision levels are generally noted by a real number with one significant decimal. This number should be (not mandatory) multiplied by 10 and used as the value of the second parameter.

Additional parameters may be present to indicate options installed in the device. These options are implementation defined, and the number and meaning of the additional parameter values will be product dependent, however, these values shall be registered with the architecture.

State Affected: None

Algorithm:

```
void device_attributes_2()
{
int n;
send_char(0x9B, host_port);    /* CSI */
send_int(PRODUCT_ID, host_port);
send_char(';', host_port);
send_int(REVISION_LEVEL, host_port);
/* Device IDs and options must be registered */
/* VT420 has no options --
for (n=1; n<NUMBER_OF_OPTIONS; n++)
    {
    if (product_options[n] == TRUE)
     {
      send_char(';', host_port);
      send_int(n, host_port);
     }
    };
--*/
send_char('c', host_port);
};    /* end procedure device_attributes_2() */
```

Known Deviations:

Terminals designed before the Terminal Interface Architecture (TIA) was developed do not implement this control. Terminals that return a first parameter value less than 60 to the primary Device Attributes (DA) will ignore the secondary Device Attributes (DA2) control.

DEVICE ATTRIBUTES (Tertiary) **DA3**
REPORT TERMINAL UNIT ID **DECRPTUI**

Level: 4

Purpose: Request terminal unit identification.

Request Format:

```
CSI          =          c
9/11         3/13       6/3
```

Description: The Tertiary Device Attributes Request (DA3) causes the terminal to report its unique terminal unit ID to the host using the DECRPTUI (Report Terminal Unit ID) sequence. The terminal unit ID consists of a string of up to 31 displayable characters that are factory set to a unique value for the specific product (DA2 registration) in which the interface is implemented.

Individual products may further restrict the length or format of the terminal unit ID as long as the identifier continues to provide a unique ID within that specific product. The format of the Terminal Unit ID shall be registered with the architecture.

Report Terminal Unit ID

Format:

```
DCS          !          |          D. . .D          ST
9/0          2/1        7/12       ?/?            9/12
```

Implementation Guideline

The VT420 terminal unit ID consists of a 32-bit number reported as four hex pairs. The first hex pair (8 bits) contains the manufacturing site code, and the remaining three hex pairs (24 bits) contain a serial number for that site. The site code and serial number together provide a unique ID for each VT420. See DEC STD 138-0 for a listing of manufacturing site codes.

The terminal unit ID is factory set and does not change unless the terminal's internal circuitry is repaired or replaced. In the VT420, the unit ID is maintained in Non-Volatile Memory (NVM) and can be rewritten in the event of NVM failure using the DECSTUI (Set Terminal Unit ID) sequence. This sequence is only recognized when the NVM checksum is invalid, or when an internal jumper on the main board is set.

Set Terminal Unit ID (DECSTUI) Format:

```
DCS          !          {          D. . .D          ST
9/0          2/1        7/11       ?/?              9/12
```

State Affected: None

Algorithm:

```
void device_attributes_3()    /* VT420 example */
{
    int n;
send_char(0x90, host_port);    /* DCS */
send_string("!|", host_port);
n = (terminal_unit_id >> 24) & 0xFF;
send_hex(n, host_port);
n = (terminal_unit_id >> 16) & 0xFF;
send_hex(n, host_port);
n = (terminal_unit_id >>  8) & 0xFF;
send_hex(n, host_port);
n = terminal_unit_id & 0xFF;
send_hex(n, host_port);
send_char(0x9C, host_port);    /* ST */
}   /* end procedure device_attributes_3() */
```

Known Deviations: None

Levels: 1, 2

Purpose: Request the device identification code.

Format:

```
ESC             Z
1/11            5/10
```

Description: The DECID control requests that the terminal report the implemented architectural conformance level and extensions. The device responds by transmitting an appropriate primary Device Attributes (DA) response.

NOTES

a. **This control is identical to a primary DA control with an omitted parameter or a parameter value of zero (0). THIS CONTROL IS RESERVED FOR FUTURE STANDARDIZATION BY ISO AND ANSI. Conforming software shall use the primary Device Attributes (DA) control for purposes of device identification.**

b. **DECID is not recognized at Level 3 or above.**

c. **Terminals that implement the VT52 emulation hardware option respond differently to this control when in VT52 mode. See *Video Systems Reference Manual - VT52 Emulation* for more information.**

State Affected: None

Algorithm:

```
/*
* DECID    ESC Z
* Ignored above Level 2
* execution is same as primary DA
*/
void id()
{
if ( (conformance_level == LEVEL_1) ||
   ( (conformance_level == LEVEL_2) )
    device_attributes_1();
};
```

Known Deviations:

The VT300 family and VT420 do not have a fully independent VT200 mode, and ignore DECID when set to conformance Level 2.

See additional deviations under Device Attributes (Primary).

SECURE RESET **DECSR**
SECURE RESET CONFIRMATION **DECSRC**

Levels: 4

Purpose: To ensure that the terminal is in a known state so that data sent to the terminal is displayed as expected, and keystrokes from the terminal transmit as expected, and to erase any host-writeable data in the terminal.

Format:

CSI	Pn	+	P
9/11	3/?	2/11	7/0

Description: Secure Reset initializes the terminal to its Power-Up state and erases all host setable state information and data. If a non-zero numeric parameter in the range 1 to 16383 is provided, the device responds with a Secure Reset Confirmation (DECSRC) control function with the same parameter upon successful completion. If Pn is omitted or zero, no response is generated. If Pn exceeds 16383, the entire sequence shall be ignored.

Secure Reset Confirmation

Format:

CSI	Pn	*	q
9/11	3/?	2/10	7/1

NOTES

1. **DECSR is recognized at all conformance levels (Level 1 or higher).**
2. **DECSR is recognized in Control Representation Mode (Display Controls Mode, CRM set) and causes the device to exit this mode. This is desirable because CRM is enabled under local control in video terminals and there is no other way to clear it from the host. If the terminal is in Display Controls Mode, DECSR shall be displayed before it is executed.**
3. **DECSR is not recognized in VT52 Mode (DECANM reset), or Tektronix 4010/4014 Emulation Mode (DECTEK). Host software must explicitly exit these modes. This is necessary because these modes do not use ANSI compatible syntaxes.**
4. **DECSR is not recognized (intercepted) in Printer Controller Mode (see Media Copy - MC). Host software must explicitly exit this mode. Secure Reset may be sent to an attached printer in Printer Controller Mode.**
5. **DECSR is buffered in the same way as other incoming data.**
6. **In a multiple-session device, DECSR only affects the session that receives the control function. At the time of this update to the standard, secure connections are only supported in single-session devices. Reconfiguring the device from a single-session to multiple-sessions causes a BREAK to be generated to alert host software.**

7. **Secure Reset need not erase data maintained in Non-Volatile Memory (NVM) as long as such data cannot be written from the host. If values for some state information can be stored under host control in NVM, then DECSR should erase this NVM (not mandatory).**
8. **DECSTUI, which sets the terminal unit ID, is an exception to the previous note since it is only recognized if the NVM checksum is invalid, or an internal jumper has been installed. In normal operation, the terminal unit ID cannot be written from the host.**

State Affected:

Secure Reset affects all host setable state similar to power-up initialization. See subhead 4, DEVICE INITIALIZATION.

The following state information is also explicitly reset:

UDKs
The UDKs are erased to be empty, or restored to the value saved in NVM if such NVM is present and not writable from the host. The UDK definitions are re-initialized even if the "UDKs Locked" state is in effect.
DRCS
Macro Definitions
clears the keyboard input silo
erases all local buffers that might contain data from host (such as paste_buffer, capture_buffer, etc...)

Secure Reset does not affect host port receive buffers, nor session management state (if present).

Algorithm:

```
void secure_reset(param)
    int param;
{
if (param <= 16383)    /* 0x3FFF */
    {
    /* exit CRM */
    control_representation_mode = INTERPRET_CONTROLS;

    /* initialize all host setable state */
    host_state_init();
    clear_udk();

    /* report reset was successful if non-zero parameter */
    if (param != 0)
     {
     send_char(0x9B, host_port);
     send_int(param, host_port);
     send_string("*q", host_port);
     };
    };
};    /* end procedure secure_reset() */
```

Known Deviations: None

SOFT TERMINAL RESET DECSTR

Levels: 2, 3, 4

Purpose: Reset soft terminal state to the default values.

Format:

```
CSI             !             P
9/11            2/1           7/0
```

Description: The DECSTR control sets the modes available to user application programs to known values. User Preference Features are not affected. DECSTR is intended for application use, whereas User Preference Feature control functions are intended for terminal management use. When the DECSTR control is received, the following state information in the terminal shall be reset as indicated below. This list defines the Soft Terminal Reset state.

State	Default
Top Margin	1
Bottom Margin	number of lines on page
Left Margin	1
Right Margin	80 (or 132 if column mode is set)
Current Rendition	normal
Current Attributes	not selective erasable
In Use Table (gl, gr)	set to current level defaults
Designated Sets	set to current level defaults
Cursor Save Buffer	set to current level defaults
Insert/Replacement Mode	reset (Replace)
Origin Mode	reset (Absolute)
Auto Wrap Mode	Auto Wrap Off (NVM if present)
Character Set Mode	8-bit Multinational (ignore NVM)
Cursor On/Off	on (ignore NVM)
Active Display	Main Display (status display is exited)
UPSS	DEC Supplemental (NVM if present)
Cursor Key Mode	reset (Cursor)
Keyboard Action Mode	reset (Unlocked)
Keypad Mode	reset (Numeric)
Keyboard Encoding Mode	character

The following User Preference Features are NOT affected by the soft reset.

Column Mode
Scroll Mode
Screen Mode
Auto Repeat Mode
Caps/Shift Lock Mode
Tab stops
Key click

The following are also NOT affected by the soft reset.

The data in the display
New Line Mode
Line Attributes
Printer Controller Mode
Auto Print Mode
Compose Mode
C1 Transmission Mode
Conformance Level
DRCS
UDK
Macro Definitions

NOTES

a. **Numeric is the default state for Keypad Mode. It is the power-up default state for the terminal. It is preferred that operating systems perform translation of keypad codes for applications software at the time data is passed to the application. This allows applications to clear the type-ahead buffer to ensure synchronization when changing keypad mode. Operating systems implementing this strategy should set Keypad Mode to Application after performing the soft reset and do the required translation.**

b. **The DECSTR control is ignored when operating at Level 1 and by Level 1 devices.**

c. **DECSTR has no affect on the Active Position.**

d. **When the terminal is in Printer Controller Mode, the DECSTR control is not executed by the terminal, but is transmitted instead to the printer port. Thus, DECSTR does not reset the terminal state unless Printer Controller Mode is explicitly reset before DECSTR is transmitted.**

e. **DECSTR is interpreted differently in printing devices to achieve the corresponding concept of resetting the device to known application state while preserving the data currently on the display (previously printed, or buffered in the printer).**

State Affected:

See subhead 4, DEVICE INITIALIZATION for type declarations.

```
/* VT420 example */
/* code extension layer state */
graphic_character_set_t  designated_graphic_sets[4];
in_use_table_t           in_use_table;
character_set_mode_t     character_set_mode;      /* NRCS */
graphic_character_set_t  upss           /* L3 or 8-bit IA */

/* character cell display state */
line_t                 top_margin;
line_t                 bottom_margin;
column_t               left_margin;   /* Horizontal Scroll */
column_t               right_margin;  /* Horizontal Scroll */
left_right_margins_mode_t  left_right_margins_mode; /* H Scroll */
active_display_t       active_display;  /* L3 or status display */
save_buffer_t          cursor_save_buffer;
character_rendition_t  current_rendition;
character_attribute_t  current_attribute;  /* Selective Erase */
origin_mode_t          origin_mode;
auto_wrap_mode_t       auto_wrap_mode;
text_cursor_enable_mode_t  text_cursor_enable_mode;
insert_replace_mode_t  insert_replacement_mode;
new_line_mode_t        new_line_mode;

/* keyboard processing state */
keyboard_action_mode_t  keyboard_action_mode;
cursor_key_mode_t       cursor_key_mode;
keypad_mode_t           keypad_mode;
keyboard_encoding_mode_t keyboard_encoding_mode;  /* L4 */
```

Algorithm:

```
void soft_terminal_reset()
{
if (conformance_level != LEVEL_1)
    {
    /* Code Extension */
    upss = DEC_SUPPLEMENTAL_G;                      /* NVM */
    in_use_table.c0 = ASCII_C;
    in_use_table.gl = ASCII_G;
    in_use_table.invoked_gl = G0;
    in_use_table.c1 = SUPPLEMENTAL_C;
    in_use_table.gr = upss;
    in_use_table.invoked_gr = G2;
    designated_graphic_sets[0] = ASCII_G;
    designated_graphic_sets[1] = ASCII_G;
    designated_graphic_sets[2] = upss;
    designated_graphic_sets[3] = upss;
    character_set_mode = EIGHT_BIT_MULTINATIONAL; /* NVM */

    /* character cell display */
    origin_mode = ABSOLUTE;
    auto_wrap_mode = WRAP_OFF;                      /* NVM */
    text_cursor_enable_mode = TEXT_CURSOR_ON;     /* NVM */
    insert_replacement_mode = REPLACE;            /* NVM */
    new_line_mode = NEW_LINE_OFF;                 /* NVM */
    active_display = MAIN_DISPLAY;
    left_right_margins_mode = FIXED;
    top_margin = 1;
    bottom_margin = lines_per_page;               /* NVM */
    left_margin = 1;
    right_margin = columns_per_page;              /* NVM */
    current_rendition.bold       = FALSE;
    current_rendition.blink      = FALSE;
    current_rendition.underscore = FALSE;
    current_rendition.reverse    = FALSE;
    current_attribute.selective_erase = FALSE;
    save_cursor();  /* initialize cursor save buffer */
    cursor_save_buffer.position.line = 1;
    cursor_save_buffer.position.column = 1;
    cursor_save_buffer.position.page = 1;

    /* keyboard processing */
    keyboard_action_mode = UNLOCKED;
    cursor_key_mode = CURSOR;                     /* NVM */
    keypad_mode = NUMERIC;
    keyboard_encoding_mode = CHARACTER;

    };
};     /* end procedure soft_terminal_reset() */
```

Known Deviations:

VT200s, VT300s, and the VT420 clear auto wrap (ignore NVM). DECterm V1 does not change the current setting of auto wrap. This was necessary for ULTRIX systems that depend on auto wrap being enabled when a new terminal is initialized.

Levels: 1, 2, 3, 4

Purpose: To cause the terminal to report the current functional status.

Request Format:

```
CSI          Ps          n
9/11         Ps          6/14
```

Report Format:

```
CSI          Ps          n
9/11         Ps          6/14
```

Description: The DSR request, with selective parameter, causes the terminal to respond with a corresponding DSR control using a parameter from the table to indicate its current functional status.

	Levels 1 - 4
5	request terminal status (command from host)
0	device ready
3	device not functional
	Level 4
?75	report status of data integrity flag (command from host)
?70	ok, no comm errors or power-up since last report
?71	error, a comm error has been detected since last report
?73	power-up, data integrity flag has not been reported since last power-up or RIS
	Multiple Sessions Extension
?85	report virtual terminal configuration (command from host)
?81	virtual terminals available through TD/SMP
?83	not configured for virtual terminals
?87	virtual terminals operating via physical lines

NOTES

1. **Some Device Status Reports (DSRs) are part of terminal management, while others pertain to presentation state or input processing. See *DEC STD 070-5 Video Systems Reference Manual - Character Cell Display* and *DEC STD 070-7 Video Systems Reference Manual - Printer Port Extension* for additional DSRs. The report status of the data integrity flag DSR is intended for application use.**
2. **On hardcopy devices DSR has often been processed asynchronously ahead of other data in the input buffer (guideline).**

3. **TD/SMP is described in the "Terminal Device/Session Management Protocol" Version 1 Architecture Specification, 13-Feb-1986.**

State Affected:

```
error_detected_flag    /* if DSR with Ps = ?75 is requested */
power_up_detected_flag
```

Algorithm:

```
void report_device_status(request, private)
    int request;
    boolean_t private;
{
switch (request)
    {
    case 5:
     if (device_status == READY)
      {
      send_char(0x9B, host_port);
      send_string("0n", host_port);
      }
     else
      {
      send_char(0x9B, host_port);
      send_string("3n", host_port);
      }
     break;
    case 75:
     if (private)
     {
     if (!error_detected_flag && !power_up_detected_flag)
      {
      send_char(0x9B, host_port);
      send_string("?70n", host_port);
      }
     else if (power_up_detected_flag)
      {
      send_char(0x9B, host_port);
      send_string("?73n", host_port);
      power_up_detected_flag = FALSE;
      }
     else
      {
      send_char(0x9B, host_port);
      send_string("?71n", host_port);
      error_detected_flag = FALSE;
      };
     };
    break;
     case 85:
 if (private)
      {
      send_char(0x9B, host_port);
      if (vt_configuration == SINGLE_SESSION)
      send_string("?83n", host_port);
     else
      if (vt_configuration == SESSIONS_ON_COMM)
      send_string("?81n", host_port);
     else
      send_string("?87n", host_port);
      };
    break;
    }
};    /* end procedure report_device_status() */
```

Known Deviations:

The VT330 and VT340 terminals do not implement the Virtual Terminal Configuration Device Status Reports.

The VT420 can report "?80 virtual terminals operating through TD/SMP", and include a second parameter that indicates the number of sessions available (when the first parameter is ?80 or ?81). It was later agreed that protocol state information should be obtained through the protocol handler, so this has not been standardized.

6 CHANGE HISTORY

6.1 REVISION 0.0 TO 0.1

The following changes were made to accommodate integration of this specification with the Video Systems Reference Manual:

1. The name of this section was changed from "Terminal Management Level 1" to "Terminal Management".
2. All references to Level 1 operation only were removed, and explicit references to Level 1 and Level 2 operation were added for all functions.
3. Some of the state names were changed to be consistent with the rest of the Video SRM.
4. The sections on Warning Bell and Text Cursor Enable Mode were removed to the section "Character Cell Display".
5. All sections on printer functions were removed to the section "Printer Port Extension".
6. The section on Hard Copy (DECHCP) was removed to the section "Sixel Graphics Extension".
7. Sections from the "Character Cell Display" section on Reset to Initial State and Device Status Report were added to this section.
8. New sections on Soft Terminal Reset and Answerback were also added to this section.
9. The algorithm for Reset to Initial State was corrected.
10. The coding of the DECID sequence was corrected.
11. Coding of other controls (most of which were removed to other section) were also corrected.
12. The coding of the Device Not Ready status response to Device Status Report was corrected.

6.2 REVISION 0.1 TO AX10

1. Removed statement that the revision level code in the secondary Device Attributes response would be zero for unreleased products.
2. Removed Answerback to an appendix. Answerback messages will not be required for conforming hardware.
3. Added Selectively Erasable Character extension to the primary Device Attributes response with parameter 6.
4. Made code changes in the primary DA algorithm. Fixed a bug in the Select Conformance Level algorithm.
5. Added note that DRCS and UDK state is cleared on Reset to Initial State in Level 2 terminals.
6. Made changes in the tables of affected state in the Soft Terminal Reset function.
7. Removed DECTST (Test) from the SRM. It will not be a required function for conforming hardware. However, Device Status Report will be required.
8. Removed Control Representation Mode from RIS.

9. Clarified that the device code returned by Primary Device Attributes will not change when the operating level changes.
10. Made New Line Mode unaffected by Soft Terminal Reset.
11. Added a note to Select Conformance Level to indicate that it is a required function in Level 2 only, but is active in Level 1 operation of Level 2 devices. Removed deviation notes on VT100 family for this function.
12. Removed Reset To Initial State as a required function, and placed it in Appendix D *Documented Exceptions*. It is not intended for use by conforming software. The RIS algorithm is now included in this section under the heading "Device Initialization".
13. Made Primary Device Attributes, Select Conformance Level, and Secondary Device Attributes a "temporal" extension to Level 1 (required in all future Level 1 implementations).
14. Added initialization of the character set table of designating escape sequences to the Soft Terminal Reset control.
15. Added setting of the Cursor Save Buffer values to the level defaults in Soft Terminal Reset.

6.3 Rev AX10 to AX11

Added "9 - NRC" to the list of Level 2 extensions.

6.4 Rev AX11 to AX12

GENERAL:

1. Removed Rev AX11 change bars. Added change bars to any change that could affect conformance or interpretation of the document or should be brought to the attention of terminal implementors or software engineers.
2. Updated Reference Standards section to include latest standards and revisions.
3. Added paragraph to "Relationship to TIA" subsection explaining distinction between functions intended for either application use, terminal management use, or communications use. Included entity functions are intended to be used by in individual function descriptions.
4. Added note to definition of Device Status Report in terminology section to distinguish DSRs used for Terminal Management versus others. Some Device Status Reports are properly part of Terminal Management, while others pertain to presentation state or input processing.
5. Added definitions for "Factory Default State", "Power-Up State", and "Tertiary Device Attributes" in terminology section.
6. Recoded all algorithms in VAX–11 C to match new Code Extension section.
7. Updated algorithms and state information throughout to include Level 3 and Level 4 Character Cell Display Service Class.

STATE DESCRIPTIONS:

8. Added note explaining the status of the algorithms at the beginning of the State Descriptions subsection. Algorithms should be viewed the same as other text.
9. Added "terminal unit ID" to device identification information available at Level 4.
10. Updated list of extensions.

INITIALIZATION:

11. Updated list of features that may be initialized from NVM to include VT300 and VT400 capabilities. Removed "keypad_mode" to reflect actual implementation in the VT100/VT200/VT300. Keypad mode is not stored in NVM.
12. Combined Control String extensions as a single enumerated type since it is not possible to be in more than one control string at a time.
13. Added character_set_mode for NRCS extension.
14. Included parser initialization from Code Extension section.
15. Added data integrity flags as defined for VT400.
16. Added virtual terminal configuration (vt_configuration) as used in VT420 to control multiple sessions.
17. Added page number, lines_per_page, columns_per_page, and cursor coupling to support Windowing Extension as implemented by VT330,VT340, and VT420.
18. Added state information for horizontal scrolling: left_margin, right_margin, left_right_ margin_mode.
19. Added area extent mode (stream versus rectangular) for rectangular area operations.
20. Added state for selecting status display type and active display.
21. Expanded keyboard state to include: keyboard_usage_mode, send_receive_mode, backarrow_key_mode, compose_key_mode, alt_key_mode, keyboard_encoding_mode (character versus position), and local controls for F1-F5.
22. Added printer_to_host per VT240, and printer_assignment for multiple sessions.
23. Added code to initialize set-up features that are not setable from the host but that affect the description of other functions.

CONTROL FUNCTIONS:

24. Clarified that higher conformance levels do not have to be strict supersets of lower levels, and that conforming software must select a known conformance level.
25. Clarified in the text that the primary DA reports extensions supported at the current operating level. If higher conformance levels subsume extensions previously reported at lower levels, these extensions must still be reported at the lower level to ensure existing software can recognize them.
26. Added concept of "Implicit Extensions" so software can recognize features defined at a higher conformance level which are not disabled at lower levels. Listed implicit extensions created by Level 2 and Level 3. Removed note that DECSCL was the only exception to rule that features defined at higher levels must be disabled at lower levels.

Implicit extensions ensure software can still recognize which features are present, and corresponds to actual VT200 and VT300 implementations.

27. Noted that DECID (ESC Z) is not recognized at Level 3 or higher.
28. Added description of Secure Reset function.
29. Updated DECSTR (Soft Terminal Reset) to reflect newer implementations as follows: Bottom margin is set to number of lines on page; left and right margins are reset; auto wrap mode is recalled from NVR; current attribute is set to not selective erasable; 8-bit multinational characters are selected; the main display is selected (status display is exited); the User Preference Supplemental Set (UPSS) is recalled from NVM; keyboard encoding mode is reset to character (VT420); Added macro definitions to list of state not affected (VT420).
30. Added Device Status Reports for data integrity and Virtual Terminal State (VT420). Noted that other DSRs are not part of terminal management.
31. Added note that DSR has been processed asynchronously ahead of other data in the input buffer on hardcopy devices (guideline).

APPENDIX A REFERENCED DOCUMENTS

A.1 EL-Class Digital Documents

EL-Class Number	Document Title
EL-00070-05	*DEC STD 070-5 Video Systems Reference Manual - Character Cell Display*
EL-00070-07	*DEC STD 070-7 Video Systems Reference Manual - Printer Port Extension*
EL-00070-0A	*Video Systems Reference Manual - VT52 Emulation*
EL-00070-0D	*Video Systems Reference Manual - Documented Exceptions*
EL-00138-00	*DEC STD 138-0 Registry of Control Functions for Character Imaging Devices*

Use VTX SMC to order copies of EL-class documents from Standards and Methods Control. Send distribution questions to JOKUR::SMC or call DTN: 234-4423.

APPENDIX B RELATED DOCUMENTS

B.1 EL-Class Digital Documents

EL-Class Number	Document Title
EL-00070-01	*DEC STD 070-1 Video Systems Reference Manual - Concepts and Conformance Criteria*
EL-00169-00	*DEC STD 169-0 DEC Standard Coded Graphic Character Sets for Hardware and Software*

Use VTX SMC to order copies of EL-class documents from Standards and Methods Control. Send distribution questions to JOKUR::SMC or call DTN: 234-4423.

B.2 Other Documents

Document Number	Document Title
ANSI X3.4 - 1986	*American National Standard Code for Information Interchange (ASCII Character Set)*
ANSI X3.41 - 1974	*American National Standard Code Extension Techniques for use with the 7-Bit Coded Character Set of the American National Standard Code for Information Interchange*
ANSI X3.64 - 1979	*Additional Controls for use with American National Standard Code for Information Interchange*
DPANS X3.134.1-1985	*8-Bit ASCII Structure and Rules*
DPANS X3.134.2-1985	*7-Bit and 8-Bit ASCII Supplemental Multinational Graphic Character Set*
ISO 2022:1986	*Information Processing / ISO 7-Bit and 8-Bit Coded Character Sets - Code Extension Techniques*
ISO 6429:1988	*Information Processing / Additional Controls for use with Character Imaging Devices*
ISO 646	*7-Bit Coded Character Set for Information Interchange*
ISO 8859-1:1987	*Information Processing / 8-Bit Single-Byte Coded Character Sets - Part 1 : ISO Latin Alphabet Nr 1;*

B.3 Ordering Information

Following are the sources for documents not available from Standards and Methods Control.

- ANSI, IEC, and ISO Documents
  American National Standards Institute (ANSI),
  1430 Broadway, New York, NY 10018
  Telephone (212) 354-3300 x479

INDEX

A

Architecture
  service class, 3

C

Conformance Levels
  definition, 2

D

DA, 17, 26
DECID, 28
DECRPTUI, 26
DECRQLP, 17
DECSCL, 22
DECSR, 29
DECSRC, 29
DECSTR, 31
DECSTUI, 26
Default
  factory, 2
  undefined, 12
Device Attributes
  control function, 17, 26
  primary, 3, 17
  secondary, 3
  tertiary, 3, 26
Device Control, 22
Device Identification, 3, 17, 26, 28
Device Status, 12, 35
Device Status Report
  control function, 35
Device Test, 12
DSR, 35

F

Factory Default State
  definition, 2

I

Identification
  device, 3
  service class, 3
Identify Terminal
  control function, 28
Implicit Extensions, 20
Initialization, 12
Initialization and Reset, 29, 31
Interface
  external, 2
  internal, 2

P

Power-Up State
  definition, 2
Primary Device Attributes
  definition, 3

R

Report Terminal Unit ID, 26
Request Locator Position
  control function, 17

S

Secondary Device Attributes
  definition, 3
Secure Reset
  control function, 29
Secure Reset Confirmation
  control function, 29
Select Conformance Level
  control function, 22
Self-Test
  definition, 2
Service Class
  identification, 3
Set Terminal Unit ID, 26
Soft Terminal Reset
  control function, 31
Software Conformance
  device identification, 28
Status Report
  definition, 2

T

Terminal Interface Architecture (TIA), 1
Terminal Management Functions, 1
  definition, 3
Tertiary Device Attributes
  definition, 3

```
+---------------------------------------------------------------------+
|                          READER COMMENTS                            |
|                                                                     |
| Your  comments  and suggestions  will help  Standards and Methods   |
| Control improve their services and documents.                       |
+---------------------------------------------------------------------+
```

Did you request this document? _____ If so, did it arrive within a satisfactory period of time? _____ Please comment.

What are your impressions of this document? Consider format, organization, completeness, readability, and illustrations.

————————————**FOLD ON THIS LINE**———————————————————————— -

Did you find technical or clerical errors in this document? If so, please specify the page number(s) and the error(s).

Are the instructions for the update package clear? _____
Was an index available? _____ If not, is one needed? _____
Do you have other suggestions for improving this document?

The following information is optional:

Name _____ Mailstop _____

Department _____ Node _____

Send your comments to JOKUR::PROJECTS, or fold, staple, and send this page through interoffice mail to:

```
+------------------------------+
|       READERS' COMMENTS      |
| STANDARDS AND METHODS CONTROL |
|          NRO4/D4             |
+------------------------------+
```